CS 361
Data Structures & Algs
Lecture 7

Prof. Tom Hayes
University of New Mexico
09-14-2010

Today

Talk about Programming Assignment 1.

Inverted Indices.

Data Structures: Arrays vs. Linked Lists.

More about Big O.

New Reading: Read section 2.5.

Reminders

- Prog #1 was due last night.
- Reading: up to sec 2.4 done.
- Written Assignment 2: problems
  1.8, 2.1, 2.2, 2.3, 2.4, 2.5*, 2.6, 2.7, 2.8+
  *:tricky. +:challenging
  Quiz: 2 next Thursday

Re Next Written Assn

Work together!

(a) in small groups, to come up with solutions

(b) online discussion, to check solutions, and test whether you know the difference between a right and wrong answer.

This assignment is long and hard! Start early!

Thoughts on P.A. #1

- Checking solution vs Finding solution
  - Unrelated tasks?
  - Relative difficulty?
  - Methods used by both?
- Object oriented design?
- Overall difficulty? Hardest tests?
- Worked together?

Resolving Proposals

Man m proposes to woman w.
w has fiance, m'.

winner: whichever of m, m' comes first in preference list of w.

Want to computer winner in O(1) time.
Can we do it?

Inverted Indices

WomenPrefs[w] lists the men by ranking.

More helpful: WomenRankings[w][m] tells where w ranks man m. 0=best, n-1=worst.

Example of an Inverted Index (wikipedia).
Application: Search engines (Google).
Pre-computers: concordances.

Preprocessing: build up this array in $O(n^2)$ time before doing any proposals.

Building an Inverted Index

WP: womens preference array
WR: womens rankings array.

```
for (w=0 to n-1)
  for (ranking=0 to n-1) {
     m = WP[w][ranking];
     WR[w][m] = ranking;
  }
```

WR[w] and WP[w] are each an inverted index of the other.

Building an Inverted Index

wife[ ]: stores a matching, (male view)
husband[ ]: inverted index (female view)

```
for (m=0 to n-1) {
   w = wife[m];
   husband[w] = m;
}
```

Inverted Indices

Making an inverted index is often a good idea. Keep it in mind!

The cost is comparable to (Θ) the cost of reading the original array.

Careful: if you modify the array, you will have to update the inverted index too!

Arrays vs Linked Lists

Operations supported. See Java API for Collections, List.

Collections Methods: add, addAll, clear, contains, containsAll, equals, hashCode, isEmpty, iterator, remove, removeAll, retainAll, size, toArray.

Collections - Subclasses

AbstractCollection, AbstractList, AbstractQueue, AbstractSequentialList, AbstractSet, ArrayBlockingQueue, ArrayDeque, ArrayList, AttributeList, BeanContextServicesSupport, BeanContextSupport,ConcurrentLinkedQueue, ConcurrentSkipListSet, CopyOnWriteArrayList, CopyOnWriteArraySet, DelayQueue, EnumSet, HashSet, JobStateReasons, LinkedBlockingDeque, LinkedBlockingQueue,LinkedHashSet, LinkedList, PriorityBlockingQueue, PriorityQueue, RoleList, RoleUnresolvedList, Stack, SynchronousQueue, TreeSet, Vector

Arrays vs Linked Lists

See wikipedia on Linked Lists for comparisons with Arrays.

Main differences: get(index), put(val, index) run in O(1) time for Array, linear time for LL. insert, delete in middle takes linear time for Array, O(1) time for LL.

Practice with big-O

Suppose $f = O(g)$ and $g = O(H)$.

Prove: $f = O(H)$.

Reasoning: Goal: $f(n) \leq C\ H(n)$.

$f = O(g)$ means: There is C_1, n_1 such that as long as $n \geq n_1$ we have $f(n) \leq C_1\ g(n)$.

$g = O(H)$ means: There is C_2, n_2 such that as long as $n \geq n_2$ we have $g(n) \leq C_2\ H(n)$.

$f(n) \leq C_1\ g(n) \leq C_1\ (C_2\ H(n)) = (C_1\ C_2)\ H(n)$

Practice with big-O

$f = O(g)$ means: There is C_1, n_1 such that as long as $n \geq n_1$ we have $f(n) \leq C_1\, g(n)$.

$g = O(H)$ means: There is C_2, n_2 such that as long as $n \geq n_2$ we have $g(n) \leq C_2\, H(n)$.

$f(n) \leq C_1\, g(n) \leq C_1\, (C_2\, H(n)) = (C_1\, C_2)\, H(n)$

Guess $C = C_1\, C_2$

$n_0 = ?$. Need: $f(n) \leq C_1\, g(n)$. From top, need $n \geq n_1$. Need: $g(n) \leq C_2\, H(n)$. Thus need $n \geq n_2$. Choose $n_0 = \max\{n_1, n_2\}$.

Practice with big-O

Suppose f = O(g) and g = O(h).

Prove: f = O(h).

Proof: f = O(g) means: There is C_1, n_1 such that as long as $n \geq n_1$ we have $f(n) \leq C_1 \, g(n)$.

g = O(H) means: There is C_2, n_2 such that as long as $n \geq n_2$ we have $g(n) \leq C_2 \, H(n)$.

Choose $C = C_1 C_2$, and $n_0 = \max\{n_1, n_2\}$.

Then

Proof: $f = O(g)$ means: There is C_1, n_1 such that as long as $n \geq n_1$ we have $f(n) \leq C_1\, g(n)$.

$g = O(H)$ means: There is C_2, n_2 such that as long as $n \geq n_2$ we have $g(n) \leq C_2\, H(n)$.

Choose $C = C_1\, C_2$, and $n_0 = \max\{n_1, n_2\}$.

Suppose $n \geq n_0$

Then

$$f(n) \leq C_1\, g(n) \leq C_1\, (C_2\, H(n)) = (C_1\, C_2)\, H(n)$$

$$= C\, H(n).$$

Thus $f = O(H)$.

Choose $C = C_1 C_2$, and $n_0 = \max\{n_1, n_2\}$.

Suppose $n \geq n_0$

Then

$f(n) \leq C_1 g(n)$ (since $n \geq n_0 \geq n_1$ and above)

$\leq C_1 (C_2 H(n))$ (since $n \geq n_0 \geq n_2$ and above)

$= (C_1 C_2) H(n)$ (arithmetic)

$= C H(n)$. (def of C)

Thus $f = O(H)$.

Choose $C = C_1 C_2$, and $n_0 = \max\{n_1, n_2\}$.

Suppose $n \geq n_0$

Then

$f(n) \leq C_1 g(n)$ (since $n \geq n_0 \geq n_1$ and above)

$\leq C_1 (C_2 H(n))$ (since $n \geq n_0 \geq n_2$ and above)

$= (C_1 C_2) H(n)$ (arithmetic)

$= C H(n)$. (def of C)

Thus $f = O(H)$.

Test your understanding

True or False:

When f, g are positive functions, "$f = O(g)$" means there is some constant C such that, for all n, $f(n)/g(n) \leq C$.

Test your understanding

True or False:

When f, g are positive functions, "f = O(g)" means there is some constant C such that, for all n, $f(n)/g(n) \leq C$.

True!

Same as $f(n) \leq C\, g(n)$.

But, what about n_0?

Test your understanding

True or False:

When f, g are positive functions, "f = O(g)" means

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = C$$

Test your understanding

True or False:

When f, g are positive functions, "f = O(g)" means

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = C$$

False. f(n)/g(n) does not have to converge to a particular value. C is only an upper bound. See board.

Test your understanding

True or False:

When f, g are positive functions, “f = Θ(g)” means

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = C$$

Test your understanding

True or False:

When f, g are positive functions, "f = Θ(g)" means

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = C$$

False. f(n)/g(n) does not have to converge to a particular value. For instance, f(n)/g(n) may oscillate between a lower bound, L, and an upper bound U.

Test your understanding

True or False:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = C \quad \text{implies f = O(g)}$$

True. Existence of this limit implies that, for large n, f(n)/g(n) is arbitrarily close to C. In particular, f(n)/g(n) is between 0 and 2C. But this implies f(n) ≤ 2C g(n), so f = O(g).

Test your understanding

True or False:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = C \quad \text{implies } f = O(g)$$

True. Existence of this limit implies that, for large n, f(n)/g(n) is arbitrarily close to C. In particular, f(n)/g(n) is between 0 and 2C. But this implies f(n) ≤ 2C g(n), so f = O(g).

Same proof shows f = Ω(g). Hence f=Θ(g)